METODY ROJU CZĄSTEK W OPTYMALIZACJI PROCESÓW TRANSPORTOWYCH I LOGISTYCZNYCH

PARTICLE SWARM METHODS IN OPTIMIZATION OF TRANSPORT AND LOGISTIC PROCESSES

Józef LISOWSKI
j.lisowski@we.am.gdynia.pl
Akademia Morska
Wydział Elektryczny
Katedra Automatyki Okrętowej

STRESZCZENIE

W artykule przedstawiono heurystyczne metody optymalizacji statycznej wykorzystujące naturalne zasady ruchu roju cząstek – ptaków, mrówek, pszczół, świetlików, nietoperzy, kryli, kukułek, mątw, karaluchów i zapylania kwiatów. Dla każdej z metod opisano cechę inteligencji roju, przedstawiono postać funkcji celu oraz podano zasadę działania algorytmu optymalizacji.

SUMMARY

The paper presents heuristic static optimization methods using the natural principles of particle swarm motion - birds, ant colony, bees, firefly, bats, krill, cuckoo search, cuttlefish, cockroaches swarm and flower pollination. For each of the methods describes the characteristics of swarm intelligence, the form of the goal function and the principle of operation of the optimization algorithm.

Słowa kluczowe: transport, logistyka, optymalizacja, informatyka
Keywords: transport, logistics, optimization, computer science

WSTĘP

Większość procesów transportowych i logistycznych posiada wiele możliwości ich praktycznej realizacji, z których wybiera się rozwiązanie optymalne spełniające jedno lub wiele kryteriów jednocześnie, na przykład minimum kosztów transportu i magazynowania, minimum czasu, minimum ryzyka awarii lub kolizji, maksimum niezawodności. Obok metod deterministycznych optymalizacji znaczną rolę zaczynają odgrywać metody heurystyczne optymalizacji statycznej, stanowiące grupę algorytmów poszukiwania rozwiązania problemów obliczeniowych w sposób pseudolosowy. Heurystyka - znajdowanie (grec. *heurisko*) to zbiór specyficznych reguł, które mogą pomóc w najlepszym rozwiązaniu problemu. Metaheurystyki jako inteligentne heurystyki bazują na analogiach do procesów ze świata rzeczywistego w zakresie fizyki i biologii, które można interpretować w kategoriach optymalizacji. Najczęściej stosowanymi metodami heurystycznymi optymalizacji statycznej

są metody grupowania, Monte Carlo, przeszukiwania, harmonii, symulowanego wyżarzania, genetyczne i ewolucyjne oraz roju cząstek.

1. CHARAKTERYSTYKA ROJU CZĄSTEK

Rój cząstek lub stado stanowi grupę osobników tego samego gatunku, rzadziej różnych gatunków zwierząt, owadów, ptaków i ryb, żyjących na określonym terytorium, związanych ze sobą mniej lub bardziej zaawansowaną forą organizacji społecznej. Łączenie się osobników w stada jest najczęściej związane z rozrodem lub poszukiwaniem pożywienia.

Algorytm optymalizacji rojem cząstek PSO (ang. *Particle Swarm Optimization*) został zaproponowany w 1995 roku przez Kennedy'ego i Eberharta (Kennedy & Eberhart, 1999). Idea algorytmu wywodzi się z naśladowania zachowania populacji żywych istot - ptaków, mrówek, pszczół, świetlików, nietoperzy, karaluchów, kryli, itp.), w których pojedynczy osobnik ma bardzo ograniczoną możliwość podejmowania decyzji i wzajemnej komunikacji (rys. 1).



Rys. 1. Przykłady osobników rojów cząstek.
Źródło: Opracowanie własne.

Całość populacji mimo braku centralnego systemu sterującego wykazuje cechy posiadania inteligencji, to jest reagowania na zmiany otoczenia i zbiorowego podejmowania związanej z tym akcji. Model numeryczny zachowania grupy obiektów traktuje populację jako rój, a każdego osobnika jako cząstkę.

W trakcie kolejnych kroków, cząstki przemieszczają się do nowych położeń, symulując adopcję roju do środowiska, czyli poszukują optimum. Algorytm wykorzystuje do poszukiwania ekstremalnej wartości funkcji przystosowania jako funkcji celu sterowania, populację ruchomych cząstek, które potrafią zapamiętywać w przestrzeni poszukiwania punkt o najlepszej wartości funkcji celu oraz przekazywać tę informację do całej lub części populacji. Na podstawie tak uzyskanego zbioru informacji poszczególne cząstki mogą zmieniać swoje położenie, dążąc do punktu o najlepszych własnościach.

Inteligencję roju cząstek można przedstawić poprzez następujące ich właściwości:

- bliskość – zdolność do obliczania czasu i odległości,
- jakość – zdolność oceny oddziaływania z otoczeniem,
- zróżnicowanie reakcji – zdolność tworzenia różnych reakcji,
- stabilność – zdolność do tworzenia zachowania o umiarkowanej reakcji na zmiany w otoczeniu,
- adaptacja – zdolność do zmian w zachowaniu, jeśli wymusza to otoczenie.

Wprowadza się następujące oznaczenia:

- populacja roju $S=\{x_1, x_2, ..., x_M\}$,
- cząstka x_i należy do N-wymiarowej przestrzeni poszukiwań D,
- położenie cząstki $x_i=\{x_{i1}, x_{i2}, ..., x_{iN}\}$,
- przemieszczenie cząstki $v_i=\{v_{i1}, v_{i2}, ..., v_{iN}\}$,
- pamięć roju P, jako położenie każdej z cząstek o najlepszej wartości funkcji celu F: $P=\{p_{i1}, p_{i2}, ..., p_{iN}\}$,
- indeks g numeru cząstki o najlepszej wartości funkcji celu w danej chwili.

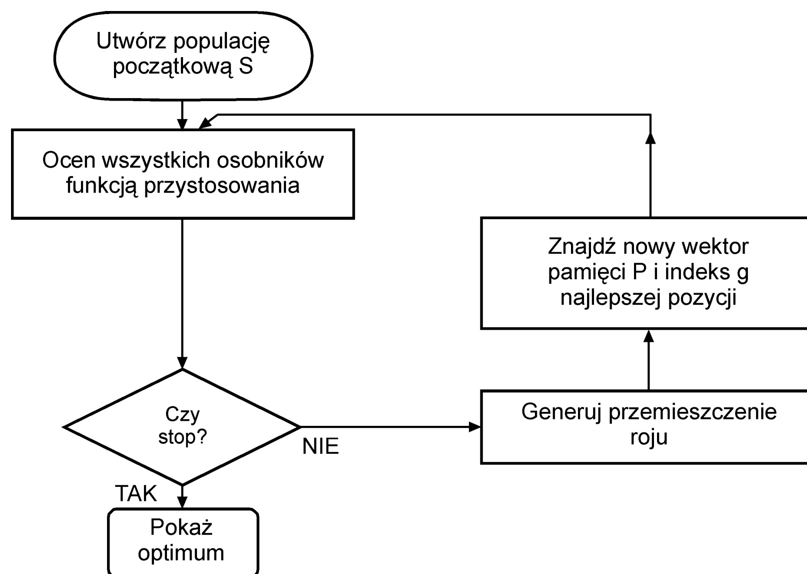
Przemieszczanie się roju cząstek można opisać dla k-tej iteracji za pomocą następujących zależności:

$$v_{ij}(k)=v_{ij}(k-1)+c_1R_{1j}\left[p_{ij}(k-1)-x_{ij}(k-1)\right]+c_2R_{2j}\left[p_{gj}(k-1)-x_{ij}(k-1)\right] \quad (1)$$

$$x_{ij}(k)=x_{ij}(k-1)+v_{ij}(k) \quad (2)$$

gdzie:
R_{1j}, R_{2j} – liczby losowe z przedziału [0;1], liczone dla każdej składowej oddzielnie,
c_1, c_2 - współczynniki wrażliwości na zmiany funkcji celu optymalizacji $F(k)$.

Zasadę działania algorytmu roju cząstek przedstawia rysunek 2.



Rys. 2. Schemat blokowy działania algorytmu roju cząstek.
Źródło: Opracowanie własne.

2. METODY ROJU CZĄSTEK

2.1. Algorytm ptasi

Algorytm ptasi PSO (ang. *Particle Swarm Optimization*) naśladuje stadne zachowanie ptaków, które komunikując się oraz wzajemnie obserwując, wymieniają pomiędzy sobą informacje, usprawniając przeszukiwanie terenu jako przestrzeni rozwiązań optymalnych (Kennedy i Eberhart, 1999). Algorytm wykorzystuje zależności (1) i (2).

2.2. Algorytm mrówkowy

Algorytm mrówkowy ACO (ang. *Ant Colony Optimization*), zaproponowany w 1999 roku przez Marco Dorigo (Dorigo, Corne i Glover, 1999), jest probabilistyczną techniką rozwiązywania problemów poprzez szukanie dobrych dróg w grafach, zainspirowany zachowaniem mrówek szukających pożywienia dla swojej kolonii. Kolonia mrówek jest zdecentralizowanym systemem rozwiązywania problemów, złożonym z wielu relatywnie prostych wzajemnie oddziaływujących jednostek, charakteryzujących się:

- samoorganizacją (ang. *self-organization*),
- łatwością przystosowywania się (ang. *flexibility*),
- odpornością (ang. *robustness*).

Elastyczność kolonii umożliwia jej adaptację do zmieniających się środowisk, a odporność oznacza funkcjonowanie kolonii nawet wtedy, gdy niektóre osobniki nie będą wypełniać swoich zadań. Algorytmy mrówkowe czerpią inspirację z obserwacji zachowania kolonii mrówek. Odkryto, że mrówki komunikują się ze sobą oraz z otoczeniem za pomocą

wydzielanych przez nie substancji chemicznych. Substancje te są nazywane feromonami (ang. pheromone). Mrówki za pomocą zostawianego przez nie na podłożu śladu feromonowego (ang. *pheromone trail*) przekazują innym mrówkom informacje.

Pierwszym problemem, do rozwiązywania którego zastosowano algorytmy mrówkowe, był problem komiwojażer *TSP* (ang. *Traveling Salesman Problem*), który stanowi zagadnienie optymalizacji ścieżki przejścia. Algorytmy mrówkowe korzystają z mechanizmu dodatniego sprzężenia zwrotnego (ang. *positive feedback*) opartego na analogii do zachowania pozostawiania na podłożu śladu i podążania za tym śladem zaobserwowanego w koloniach mrówek. Do tego celu stosowany jest tzw. wirtualny ślad feromonowy (ang. *virtual pheromone trail*). Za pomocą tego mechanizmu dobre rozwiązania są utrzymywane w pamięci, dzięki czemu mogą służyć do uzyskiwania jeszcze lepszych rozwiązań. Trzeba jednak uważać, żeby wzmacniając dobre, ale nie bardzo dobre rozwiązania nie doprowadzić do przedwczesnej zbieżności algorytmu do minimum lokalnego, czyli do tzw. stagnacji (ang. *stagnation*). Aby temu zapobiec stosowany jest pewien rodzaj ujemnego sprzężenia zwrotnego (ang. *negative feedback*), które polega na wyparowywaniu śladu feromonowego (ang. *pheromone evaporation*).

Wykorzystanie algorytmu mrówkowego ACO do wyznaczania bezpiecznej trajektorii statku w sytuacji kolizyjnej opracowała A. Lazarowska (Lazarowska, 2015). Obliczenia bezpiecznej trajektorii statku własnego przy wykorzystaniu algorytmu mrówkowego składają się z trzech następujących etapów:

- inicjalizacja danych,
- konstruowanie rozwiązań,
- aktualizowanie śladów feromonowych (rys. 3).

Trasę statku od punktu początkowego wp_0 do punktu końcowego wp_e dzieli się na k etapów. Spotkane statki reprezentowane są przez sześciokątne domeny, których własny statek nie może przekroczyć (rys. 4). Prawdopodobieństwo wyboru kolejnego wierzchołka przez mrówkę wynosi:

$$p_{wp_{ij}}(t) = \frac{\left[\tau_{wp_j}(t)\right]^{\alpha} \left(\eta_{wp_{ij}}\right)^{\beta}}{\sum\limits_{l \in wp_i} \left[\tau_{wp_l}(t)\right]^{\alpha} \left(\eta_{wp_{il}}\right)^{\beta}} \tag{3}$$

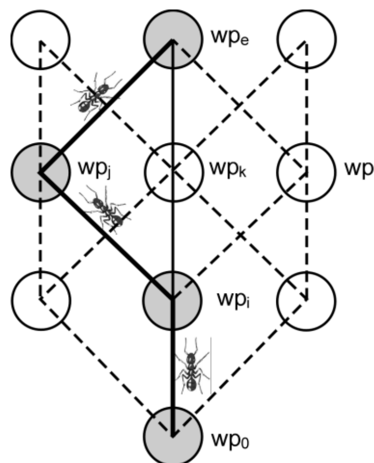
gdzie:

$\tau_{wpj}(t)$ - wartość śladu feromonowego na wierzchołku *j*,

η_{wpij} - pewna informacja heurystyczna, zwana widocznością (ang. *visibility*), odwrotność odległości pomiędzy aktualnym wierzchołkiem *i* a sąsiadującym wierzchołkiem *j*.



Rys. 3. Algorytm mrówkowy wyznaczania bezpiecznej trajektorii statku.
Źródło: Lazarowska, 2015.



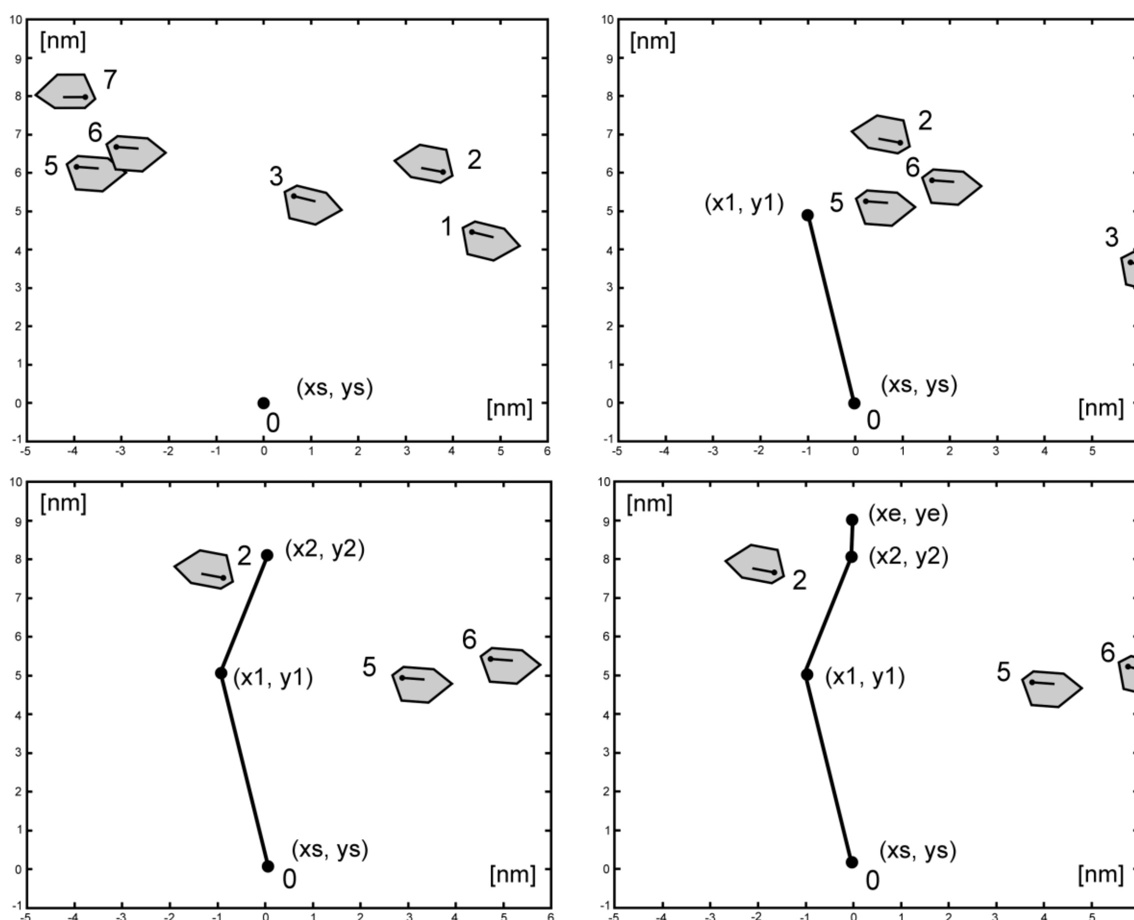
Rys. 4. Przykładowa trasa wybrana przez mrówkę.
Źródło: Lazarowska, 2015.

Jeżeli współczynnik α=0, to najbardziej prawdopodobny jest wybór najbliższego wierzchołka, jeżeli zaś β=0, wówczas działa tylko wzmocnienie feromonowe, co prowadzi do niezadawalającego rozwiązania. Gdy wszystkie mrówki w danej iteracji algorytmu zakończą

konstruowanie swojej trasy, następuje proces aktualizowania śladu feromonowego, który składa się z dwóch etapów:

- wyparowywanie feromonu (ang. *pheromone evaporation*), które polega na zmniejszeniu śladu feromonowego na wszystkich wierzchołkach grafu o pewną stałą wartość, gdzie ρ oznacza współczynnik wyparowywania śladu feromonowego,
- osadzanie śladu feromonowego (ang. *pheromone deposit*), pewna wartość śladu feromonowego dodawana jest do wszystkich wierzchołków grafu należących do tras skonstruowanych przez mrówki w danej iteracji.

Na rysunku 5 przedstawiono przykład wyznaczania bezpiecznej trajektorii własnego statku za pomocą algorytmu mrówkowego.



Rys. 5. Rozwiązanie sytuacji spotkania z 7 spotkanymi statkami w Cieśninie Kattegat.
Źródło: Lazarowska, 2015.

2.3. Algorytm pszczeli

Żerowanie roju miodnych pszczół naśladuje populacyjny algorytm poszukiwań BBA (ang. *Base Bees Algorithm*), opracowany przez Duc Truong Pham w 2005 roku (Pham i inni, 2005).

Wszystkie gatunki pszczół żyją w społeczeństwach, zakładając duże kolonie, zależnie od gatunku liczące od kilku do 20-80 tysięcy osobników. Życie rodzin jest koordynowane za pośrednictwem feromonów, z podziałem ról społecznych.
Algorytm BBA wykonuje lokalne przeszukiwania w połączeniu z mechanizmami losowymi. Kolonia pszczół miodnych może poruszać się na duże odległości powyżej 14 kilometrów w wielu kierunkach jednocześnie, korzystać z wielu źródeł żywności i rozwijać się gromadząc żywność. Tereny kwiatów zawierające znaczne ilości nektaru i pyłków gromadzone są przy mniejszym wysiłku wielu pszczół. Proces poszukiwania pożywienia rozpoczyna się od wysłania z kolonii pszczół zwanych zwiadowcami, w obiecujące tereny kwiatowe. Pszczoły zwiadowcy poruszają się losowo z jednego miejsca na drugie. Podczas sezonu zbierania pożywienia, kolonia pszczół kontynuuje eksplorację, utrzymując pewien procent osobników do przeprowadzania zwiadu. Po powrocie do ula, pszczoły wykonujące zwiad są oceniane. Powyżej pewnego progu jakości, mierzonej jako połączenie niektórych składników takich, jak zawartość cukru i złożenie nektaru lub pyłku, przechodzą do tańca *waggle dance*. Taniec pszczół, jako ich mowa, jest konieczny dla ich komunikacji w kolonii i zawiera trzy następujące informacje: kierunek ruchu, odległość od ula i ocenę jakości źródła pożywienia. Pozwala to na wysłanie swoich pszczół w dokładne miejsce położenia kwiatów. Po wykonaniu tańca, pszczoła zwiadowca, powraca na tereny kwiatowe z pszcołami rekrutami czekającymi w ulu. Podczas zbioru pożywienia, pszczoły monitorują jego poziom i ta informacja jest niezbędna do podjęcia decyzji następnego tańca po powrocie do ula.
Algorytm wymaga założenia wartości następujących parametrów:
- liczba pszczół zwiadowców *n*,
- liczba miejsc wybranych z odwiedzonych *m*,
- liczba najlepszych obszarów *e* dla *m* odwiedzanych miejsc,
- liczba rekrutów *nep* dla najlepszych obszarów *e*,
- liczba rekrutów *nsp* dla innych *m-e* wybranych obszarów,
- początkowy rozmiar obszarów *ngh* oraz sąsiedztwa.

Istnieją następujące modyfikacje podstawowego algorytmu BBA:

- algorytm ABC (ang. *Artificial Bee Colony*), zaproponowany przez Karaboga w 2005 roku, w którym symuluje się inteligentne zachowanie zbierania pożywienia przez rój pszczół, a kolonia pszczół składa się z dwóch grup – pszczół robotnic i pszczół bezrobotnych, jako obserwatorów i zwiadowców. Ilość nektaru ze źródła pożywienia odpowiada jakości rozwiązania. Liczba pracujących pszczół jest równa liczbie źródeł żywności. Sztuczna pszczoła jako obserwator wybiera źródło pożywienia na podstawie prawdopodobieństwa *p*.

- algorytm MHBO (ang. *Marriage in Honey-Bees Optimization*) zaproponowany przez H.A. Abbas w 2001 roku, opierający się na mariażu królowej z trutniami. Królowe reprezentują potencjalne optymalne rozwiązanie. Matki odbywają loty godowe z trutniami, przez co powstaje potomstwo, następnie potomstwo zostaje ulepszone przez robotnice za pomocą określonej metaheurystyki. Najgorsze matki są zastępowane najlepszym potomstwem.

2.4. Algorytm świetlika

W 2008 roku prof. Xin-She Yang opracował algorytm świetlika FA (ang. *Firefly Algorithm*), zainspirowany zachowaniem społecznym świetlików, owadów z rodziny Lampyridae, w Polsce określanych mianem robaczków świętojańskich, których fenomenem jest bioluminescencyjna komunikacja (Yang, 2008).

Algorytm służy do rozwiązania zadania optymalizacji ciągłej z ograniczeniami problemu minimalizacji funkcji kosztu $F(x)$, poprzez znalezienie takiej wartości zmiennej x^*, że zachodzi:

$$F\left(x^*\right) = \min_{x \in S} f\left(x\right) \tag{4}$$

przy czym: $S \in \Re^n$.

Algorytm reprezentuje trzy zasady:

- świetliki są bezpłciowe, oznacza to, że dowolny świetlik może być atrakcyjny dla innego świetlika,
- atrakcyjność świetlika jest proporcjonalna do jego jasności, która maleje wraz ze wzrostem odległości między świetlikami, jeśli wszystkie świetliki są tak samo atrakcyjne to poruszają się losowo,
- jasność świecenia świetlika jest determinowana poprzez dopasowanie do funkcji celu.

Załóżmy, że istnieje rój m agentów (świetlików) rozwiązujących iteracyjnie problem optymalizacji, przy czym x_i reprezentuje rozwiązanie dla świetlika i w iteracji k, a $f(x_i)$ oznacza jego koszt. Każdy świetlik posiada wyróżniającą go atrakcyjność β, która określa jak silnie przyciąga on innych członków roju. Jako atrakcyjność świetlika powinno się użyć malejącej funkcji odległości $r_j=d(x_i,x_j)$ do wybranego świetlika j:

$$\beta = \beta_0 e^{-\gamma r_j^2} \tag{5}$$

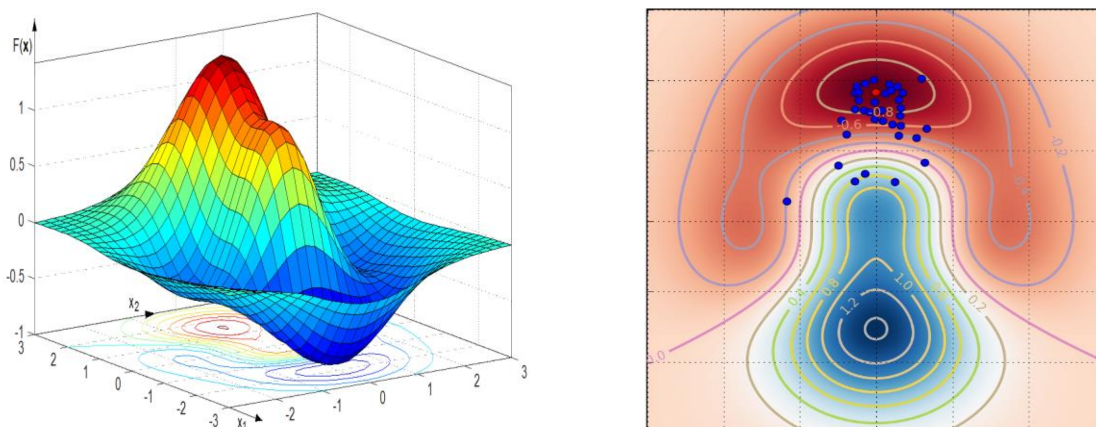
gdzie:
β_0 - maksimum atrakcyjności, γ - współczynnikiem absorpcji.

Każdy uczestnik roju jest charakteryzowany przez natężenie światła I_i, które może być bezpośrednio wyrażone jako odwrotność wartości funkcji kosztu $f(x_i)$. Początkowo wszystkie

świetliki są rozmieszczone w przestrzeni S losowo bądź z użyciem pewnej deterministycznej strategii. Aby efektywnie eksplorować przestrzeń rozwiązań przyjmuje się, że każdy świetlik i zmienia swoje położenie iteracyjnie, biorąc pod uwagę dwa czynniki: atrakcyjność innych członków roju o większej świetlności $I_j > I_i$, $\forall j=1,...m$, $j \neq i$, która maleje wraz z odległością oraz krok losowy u_i. Dla najjaśniejszego świetlika stosuje się jedynie wyżej wymieniony krok losowy.

Ogólną strukturę algorytmu można przedstawić w trzech następujących etapach:

1) Inicjalizacja parametrów algorytmu (n, β_0, γ, α, liczba iteracji). Zdefiniowanie funkcji celu $f(x)$. Wygenerowanie populacji świetlików. Intensywność światła i-tego świetlika jest określona przez funkcję celu $f(x_i)$.
2) Dopóki nie jest spełniony warunek stopu (zadana liczba iteracji lub brak ruchu świetlików) dla wszystkich n świetlików należy:
   - jeżeli ($I_j > I_i$), to wykonać ruch świetlika i w kierunku świetlika j,
   - wyznaczyć atrakcyjność,
   - znaleźć nowe rozwiązanie i uaktualnić intensywność światła.
3) Spełnienie warunku stopu, wskazanie świetlika z najwyższą intensywnością światła, a następnie wizualizacja wyników (rys. 6).



Rys. 6. Wykres funkcji celu $F(x_1,x_2)$ oraz zobrazowanie poszukiwania optimum funkcji celu $F^*(x_1^*, x_2^*)$ za pomocą algorytmu świetlika.

Źródło: Opracowanie własne.

2.5. Algorytm kukułki

Algorytm wykorzystujący zwyczaje lęgowe kukułki CS (ang. *Cuckoo Search*), zaproponowany w 2009 roku przez Xin-She Yang i Suash Deb, naśladuje zachowanie niektórych gatunków kukułki, które wykorzystują gniazda innych ptaków do wylęgu jaj i wychowywania piskląt. Zasada algorytmu CS:

- trzy uproszczone reguły:
  - każda kukułka składa jedno jajo w danej chwili w losowo wybranym gnieździe,
  - najlepsze gniazda z wysoką jakością jaj będą przenoszone na następne pokolenia,
  - liczba dostępnych gniazd jest stała, a jajo podrzucone przez kukułkę jest wykryte z prawdopodobieństwem p_a [0,1],
- ptak gospodarz może albo wyrzucić jajko z gniazda lub zrezygnować z niego i zbudować zupełnie nowe,
- ostatnie założenie może być oszacowaniem pewnej ilości gniazd, które zastępowane są przez nowe gniazda z nowymi losowymi rozwiązaniami p_a,
- jakość lub przydatność rozwiązania może być proporcjonalna do wartości funkcji celu *F*,
- jajo w gnieździe reprezentuje rozwiązanie,
- jajo kukułki jest nowym rozwiązaniem,
- nowe i potencjalnie lepsze rozwiązanie (jajo kukułki) zastępuje słabe rozwiązanie w gnieździe,
- algorytm może być rozszerzony do wersji gdzie wiele jaj reprezentuje zbiór rozwiązań,
- proste podejście – każde gniazdo ma jedno jajo.

Zachowanie lotu wielu zwierząt i owadów można opisać za pomocą lotów Lévy'ego, mających charakter błądzenia losowego, w którym długość kroku jest wyznaczana w oparciu o rozkład „długiego ogona". Lot Lévy odnosi się do francuskiego matematyka Paula Pierre Lévy (Lisowski, 2017).

Szczegółowy opis algorytmu CS:

- tworzenie nowego rozwiązania: $x(t+1)=x(t)+\alpha u$

  gdzie: $\alpha > 0$ - wielkość kroku związana z wielkością dziedziny parametrów,

  u - liczba wylosowana z rozkładem Lévy'ego, o nieskończonej wariancji i wartości średniej, $u \sim t^{-\lambda}$, $1<\lambda\leq 3$,
- niektóre nowe rozwiązania powinny być generowane na podstawie lotu Lév'iego wokół najlepszego rozwiązania, co powoduje przyspieszenie lokalnych poszukiwań,
- znaczna część nowych rozwiązań powinna być generowana w odległych regionach niż obecne najlepsze rozwiązanie,
- pozwoli to upewnić się, że system nie będzie uwięziony w lokalnym optimum.

Na rysunku 7 zobrazowano poszukiwanie za pomocą algorytmu kukułki minimum funkcji celu *F* na przykładzie funkcji Michalewicza:

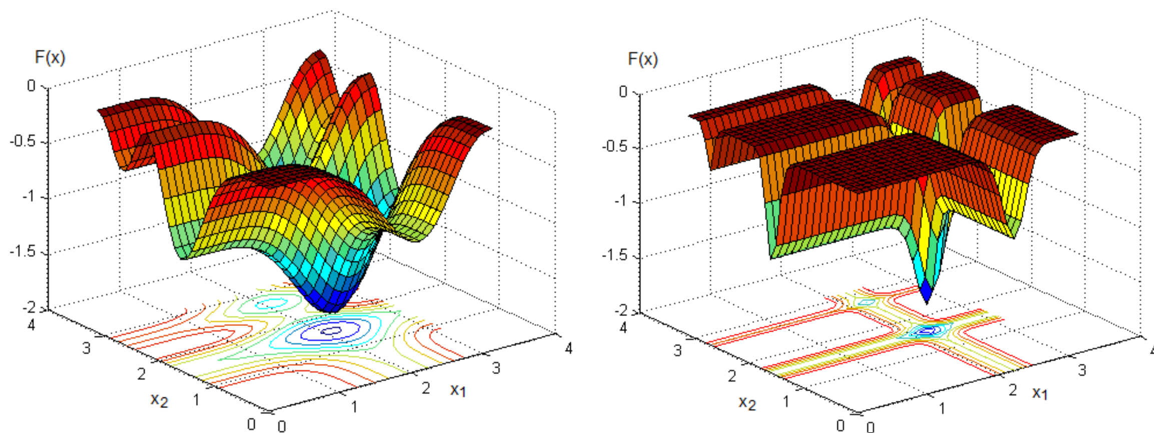
$$F(x) = -\sum_{i=1}^{n} \sin x_i \left(\sin\left(\frac{i x_i^2}{\pi} \right) \right)^{2m}, \quad 0 \le x_i \le \pi \tag{6}$$

gdzie:
$n \in Z_+$ - wymiar funkcji, $m \in Z_+$ - parametr funkcji, najczęściej m=10.

Funkcja Michalewicza to funkcja multimodalna, stosowana jako narzędzie do testowania algorytmów optymalizacyjnych. Obszar testowy jest zwykle ograniczony hipersześcianem $0 \le x_i \le \pi$; i=1,.., n. Funkcja w tym obszarze posiada n! minimów lokalnych, gdzie n to wymiar funkcji. Parametr m jest odpowiedzialny za kąt nachylenia dolin i krawędzi. Trudność poszukiwania minimum wzrasta proporcjonalnie do wzrostu parametru m. Dla dużych wartości m minimalizację funkcji można porównać do szukania igły w stogu siana, gdyż wartości funkcji są stałe poza wąskimi dolinami. Z rosnącym parametrem m doliny zwężają się i poszukiwanie minimum jest coraz trudniejsze.

W przestrzeni dwuwymiarowej (n=2) funkcja Michalewicza przyjmie postać:

$$F(x_1, x_2) = -\sin x_1 \left(\sin\left(\frac{x_1^2}{\pi} \right) \right)^{2m} - \sin x_2 \left(\sin\left(\frac{2x_2^2}{\pi} \right) \right)^{2m}, \quad 0 \le x_1, x_2 \le \pi \tag{7}$$



Rys. 7. Wykres funkcji Michalewicza dla m=1 (z lewej) - minimum wyznaczone za pomocą algorytmu kukułki: $F^* = -1{,}8409,\ x_1^* = 2{,}073,\ x_2^* = 1{,}572$ oraz dla m=10 (z prawej): $F^* = -1{,}8013,\ x_1^* = 2{,}204,\ x_2^* = 1{,}572$.

Źródło: Opracowanie własne.

2.6. Algorytm nietoperza

Nietoperze to jedyne ssaki zdolne do lotu, żyjące w stadach i prowadzące nocny tryb życia, używają echolokacji wysyłając ultradźwięki, które do nich wracają, co stanowi podstawę algorytmu BA (ang. *Bat Algorithm*) opisanego przez Xin-She Yang w 2010 roku.

Populację nietoperzy reprezentuje *i* osobników, znajdujących się w pozycjach x_i, poruszających się z prędkościami v_i i generujących impulsy z częstotliwościami f_i. Nietoperze latając losowo, zmieniając długość fal i głośność szukają zdobyczy, mogą dostosować tempo emisji impulsów do odległości od swojego zadania (Yang, 2010).

2.7. Algorytm karalucha

Trzy zachowania karaluchów jako insektów: tworzenie roju (ang. *swarming*), rozpraszanie (ang. *dispersing*) i zachowanie bezwzględne (ang. *ruthless behaviour*) wykorzystuje algorytm CSO (ang. *Cockroach Swarm Optimization*), opisany przez L. Cheng, Z.B. Wang, Y.H. Song oraz A.H. Guo w 2011 roku (Cheng, Wang, Song i Guo, 2011).
W pierwszym zachowaniu każdy karaluch szuka wśród widocznych dla niego innych karaluchów takiego, którego rozwiązanie jest lepsze niż jego własne. Jeżeli znajdzie takiego, to zmienia swoje rozwiązanie wykonując krok w stronę karalucha o lepszym rozwiązaniu o długości losowanej z przedziału [0, step], gdzie *step* jako maksymalna długość kroku jest ustalanym parametrem na początku algorytmu. Jeżeli wśród wszystkich widocznych dla siebie karaluchów osobnik nie znajdzie lepszego rozwiązania, to wykonuje analogiczny krok w stronę aktualnego globalnego najlepszego rozwiązania. W drugim zachowaniu każdy karaluch wykonuje losowy krok w przestrzeni rozwiązań, co w algorytmie wprowadza element losowości. W trzecim zachowaniu silniejszy osobnik zjada słabszego, co w algorytmie przejawia się przyjęciem przez losowego karalucha z populacji wartości aktualnego globalnego optimum.

2.8. Algorytm zapylania kwiatów

Proces zapylania roślin kwiatowych stał się inspiracją do opracowania przez Xin-She Yang w 2012 roku algorytmu FPA (ang. *Flower Pollination Algorithm*), który reprezentuje następujące reguły:

- biotyczne i krzyżowe zapylanie uważa się za proces zapylania globalnego, w którym pyłki przenoszą zapylacze wykonujące loty Lévy,
- abiotyczne i samozapylenie są uważane za lokalne zapylanie,
- stałość kwiata można uznać jako prawdopodobieństwo reprodukcji, które jest proporcjonalne do podobieństwa dwóch kwiatów zaangażowanych,
- miejscowe i globalne zapylanie jest kontrolowane przez prawdopodobieństwo przełączania $p \in [0,1]$, ze względu na fizyczną bliskość innych czynników, takich jak wiatr, lokalne zapylanie może mieć znaczny udział w *p* w ogólnej aktywności zapylania (Yang, 2012).

Powyższe reguły prowadzą do następujących zależności:

$$x_i^{t+1} = x_i^t + L\left(x_i^t - g*\right)$$
$$x_i^{t+1} = x_i^t + \varepsilon\left(x_i^t - x_k^t\right) \quad (8)$$

gdzie:
x_i^t - wektor rozwiązania, g* - bieżące, podczas iteracji, znalezione najlepsze rozwiązanie,

Prawdopodobieństwo przełączania między dwoma równaniami podczas iteracji wynosi *p*. Ponadto, ε jest liczba losową pochodzącą z rozkładu normalnego. *L* jest wielkością kroku wynikającą z rozkładu Lévy. Loty Lévy za pomocą kroków Lévy to silny krok losowy, ponieważ w tym samym czasie mogą być realizowane zarówno globalne, jak i lokalne możliwości wyszukiwania. W przeciwieństwie do standardowego poszukiwania rozwiązania, loty Lévy przedstawiają długie skoki, które pozwalają algorytmowi wyskoczyć z lokalnych dolin. Kroki Lévy są zgodne z następującym przybliżeniem:

$$L \cong \frac{1}{s^{1+\beta}} \quad (9)$$

gdzie: β - wykładnik Lévy.

Może to być trudne do ustalenia prawidłowych kroków Lévy, a prostym sposobem generowania lotów Levy jest użycie dwóch normalnych rozkładów *u* i v przez transformację:

$$s = \frac{u}{|v|^{1+\beta}} \quad (10)$$

gdzie:
$u \cong N\left(0, \sigma^2\right), v \cong N(0,1)$, σ - jest funkcją β.

2.9. Algorytm kryla

Algorytm KH (ang. *Krill Herd*), opisany w 2012 roku przez Amir Hossein Gandomi i Amir Hossein Alavi, jest oparty na symulacji zachowań stadnych osobników kryla (Gandomi and Alavi, 2012). Minimalne odległości każdego kryla od żywności i od największej gęstości stada uważane są za obiektywną funkcję ruchu kryla. Zależne od czasu, położenie osobników kryla jest sformułowane przez trzy główne czynniki:

- ruch indukowany przez obecność innych osobników,
- aktywność karmienia,
- przypadkową dyfuzję.

Algorytm KH korzysta z modelu Lagrangiana w następujący sposób:

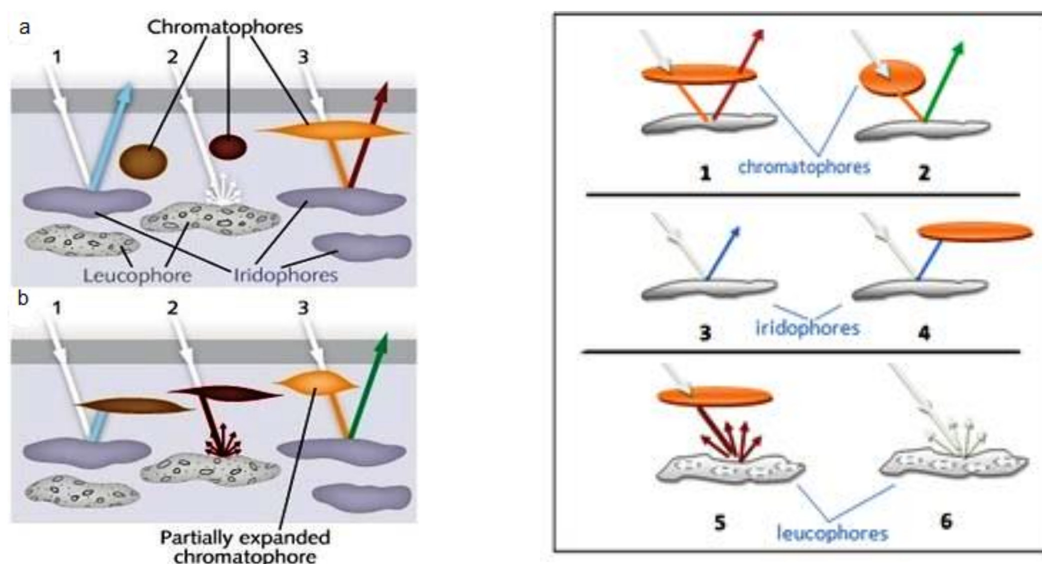
$$\frac{dX_i}{dt} = N_i + F_i + D_i \quad (11)$$

gdzie:

N_i - jest ruchem wywołanym przez inne osobniki kryla,
F_i - jest ruchem żerowania,
D_i - jest fizycznym rozproszeniem *i* osobników kryla.

2.10. Algorytm mątwy

Algorytm CFA (ang. *Cuttlefish Algorithm*), zaproponowany przez Adel Sabry Eesa, Adnam Mohsin Abdulazeez Brifcani i Zeynep Orman w 2013 roku, jest inspirowany przez zmianę środowiskową koloru skóry mątwy (Eesa, Brifcani i Orman, 2013). Algorytm posiada dwa globalne i dwa lokalne wyszukiwania oraz uwzględnia dwa główne procesy: odbicie (*reflection*) i widoczność (*visibility*). Proces odbicia symuluje mechanizm odbicia światła, a widoczność symuluje widoczność dopasowanych wzorców. Te dwa procesy są wykorzystywane jako strategie wyszukiwania, aby znaleźć globalne rozwiązanie optymalne. Formuła znalezienia nowego rozwiązania (*newP*) przy użyciu odbicia i widoczności jest następująca: *New P=Reflection+Visibility*. Rysunek 8 przedstawiający mątwę, obrazuje trzy struktury skóry - chromatofory, irydofory i leukofory, na dwóch przykładach (a,b) i trzy odrębne ślady promieniowania (1,2,3) pokazujące oryginalne środki, dzięki którym mątwy mogą zmieniać kolor odbijając światło. Algorytm imituje pracę tych trzech komórek poprzez zmianę kolejności sześciu przypadków.



Rys. 8. Fazy zmiany zabarwienia mątwy (z lewej) oraz zmiana kolejności sześciu przypadków zabarwienia mątwy (z prawej).
Źródło: Eesa, Brifcani i Orman, 2013.

Algorytm CFA dzieli populację na 4 grupy (G1, G2, G3 i G4). Dla grupy G1 algorytm stosujący przypadek 1 i 2 (interakcje między chromatoforami i irydoforami) w celu wytworzenia nowych rozwiązań. Te dwa przypadki są używane jako globalne wyszukiwania. Dla grupy G2 algorytm wykorzystuje przypadek 3 (odbicie irydoforytów) oraz przypadek 4

(interakcja między irydoforami i chromatoforami) w celu wytworzenia nowych rozwiązań jako lokalnego wyszukiwania. Podczas gdy w grupie G3 interakcja leukoforów i chromatoforów (przypadek 5) jest wykorzystywana do wytwarzania rozwiązań wokół najlepszych rozwiązań (wyszukiwanie lokalne). Następnie dla grupy G4, przypadek 6 (odbicie leukoforów) wykorzystuje się jako globalne poszukiwanie, odzwierciedlając wszelkie przychodzące światło, bez jakiejkolwiek modyfikacji.

3. PODSUMOWANIE

Algorytmy heurystyczne cechuje znaczny potencjał rozwiązywania trudnych problemów optymalizacyjnych, intuicyjna inspiracja mechanizmów biologicznych oraz możliwość ich łączenia z metodami klasycznymi optymalizacji.

Szeroki zakres zastosowań algorytmów optymalizacji rojem cząstek stanowi perspektywiczny przedmiot dalszych badań naukowych w zakresie rozwiązywania złożonych problemów optymalizacyjnych występujących w transporcie i logistyce, zwłaszcza morskiej.

LITERATURA

Cheng, L., Wang, Z.B., Song, Y.H., Guo, A.H. (2011). Cockroach swarm optimization algorithm for TSP. *Advanced Engineering Forum*, 1, 226-229.

Dorigo, M., Corne, D. and Glover, F. (1999). *Swarm intelligence: from natural to artificial systems*. Oxford: Oxford University Press.

Eesa, A.S., Brifcani, A.N.A. and Orman, Z. (2013). Cuttlefish algorithm – a novel bio-inspired optimization algorithm. *International Journal of Scientific & Engineering Research*, 4(9), 1978-1986.

Gandomi, A.H. and Alavi, A.H. (2012). Krill herd: a new bio-inspired optimization algorithm. *Commun Nonlinear Sci Numer Simulat*, 17, 4831-4845.

Kennedy, J. and Eberhart, R.C. (1999). The particle swarm: social adaptation in information-processing systems. In D. Corne, M. Dorigo, and F. Glover (eds.), *New ideas in optimization*. London: McGraw-Hill.

Lazarowska, A. (2015). Swarm intelligence approach to safe ship control. *Polish Maritime Research*, 22(4), 34-40.

Lisowski, J. (2017). *Metody optymalizacji*. Gdynia: Wydawnictwo Akademii Morskiej, 133-165.

Pham, D.T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S. and Zaidi, M. (2005). *The bees algorithm*. Technical Report: MEC 0501, Manufacturing Engineering Centre, Cardiff University.

Yang, X.S. (2008). *Nature-inspired metaheuristic algorithms*. Luniver Press.

Yang, X.S. and Deb, S. (2009). *Cuckoo search via Lévy flights*. World Congress on Nature & Biologically Inspired Computing, NaBIC 2009, IEEE Publications, 210-214.

Yang, X.S. (2010). *A new metaheuristic bat-inspired algorithm*. In: Nature inspired cooperative strategies for optimization, NISCO 2010. Studies in Computational Intelligence, 284, 65-74.

Yang, X.S. (2012). *Flower pollination algorithm for global optimization*. Lecture Notes in Computer Science, 7445, 240-249.